Santa Clara High Technology Law Journal

Volume 24 | Issue 4 Article 4

2008

Outsource Software Development and Open Source: Coming of Age in the 2000s

Heather Meeker

OUTSOURCE SOFTWARE DEVELOPMENT AND OPEN SOURCE: COMING OF AGE IN THE 2000S

Heather Meeker[†]

Abstract

This article gives a brief overview of certain issues that outsource development service providers should consider to ensure open source software license compliance. Among other topics, this article considers various types of licenses, development tools, and addresses the "developer problem."

† J.D., Chair IP/IT Licensing and Transactions Group, Greenberg Traurig LLP.

I. INTRODUCTION

First they ignore you, then they laugh at you, then they fight you, then you win.

–Mohandas Gandhi

The non-violent revolution of open source software, which has been re-working the world's software licensing landscape for the last 15 years, is on the cusp of maturity. It is fitting that Gandhi's famous quotation so accurately describes the adoption of open source by the technology industry,[1] because outsource software development providers, particularly in emerging economies like India, China, and Eastern Europe, are quickly internalizing the U.S. software industry's best practices for dealing with open source legal risks.

By all indications, open source is somewhere between the fighting stage and the winning stage. Certainly, it is no longer ignored; in the United States, conferences on open source and open source licensing are ubiquitous. Steve Ballmer's infamous description of free software as "communist"[2] is already eight years in the past. At this point, best practices in the technology sector include policies and procedures for vetting the inclusion of open source software in commercial products.[3] Some companies, particularly in the United States, have become quite sophisticated in this regard. However, communicating software development standards to outsource providers is difficult in the best case—facing, as it does, the challenges of translation errors and cultural disconnects. When it comes to open source, this is even more difficult, because outsource providers are being required to come up to speed quickly to follow U.S.-driven policies. U.S. companies have developed best practices to inoculate their products against open source licensing risks gradually and organically. Now, outsource providers are being tasked with following policies that differ from client to client, and add significant overhead to software development activity.

So now, in the late 2000s, outsource providers are scrambling to meet their clients' requirements for best development practices

---

1. My thanks to Arunesh Dayal of Red Hat, who used this quotation at his presentation at the ITechLaw conference in Mumbai in February 2008.

2. Which may be apocryphal, but see Graham Lea, MS' Ballmer: Linux is communism (July 31, 2000), http://www.theregister.co.uk/2000/07/31/ms_ballmer_linux_is_communism/.

3. *See* Karen Faulds Copenhaver, *Compliance with Open Source Software Licenses*, LICENSING J. (Sept. 1, 2006).

relating to open source. But those clients themselves, for the most part, have only recently put their own house in order after a long period of denial and ignorance of open source licensing risks. Outsource clients today are like parents telling their outsource developer teenage children to straighten up and fly right—which is like a thirty year old father telling his fifteen year old son to stop partying, when it was not so long ago that the father was doing exactly the same partying himself.

II. GIGO—THE ETERNAL INFORMATION PROBLEM

The vast majority of time spent resolving issues in open source licensing is the gathering of information—or more precisely, the vetting of the information. Anyone can collect inaccurate or incomplete information, quickly and easily. As every programmer knows, bad data means bad results—garbage in, garbage out. Good record keeping is the answer, of course, but it is a harder answer to implement than it seems.

Outsource providers will get significant benefit for their efforts by streamlining their information tracking systems to record and update open source licensing terms for software used in development. The information typically needed is:

- Software package name
- Version
- License (including version)
- Download URL
- Dependencies
- Whether the code has been modified
- Copyright notices

The version of the software is necessary, because open source projects do change their licenses over time, so the version of the software may help to verify licensing terms. The version of the license is necessary because subsequent versions of licenses can be quite different in effect (such as the many changes introduced in GPL version 3 and Apache Software License version 2.0). The download URL is helpful to verify the licensing terms. Finally, dependencies will help identify software that may have been omitted from the records, and help address compliance issues with copyleft licenses like GPL.

Simply keeping the right information will help outsource providers tremendously to meet the needs of their customers. In fact, some customers will prefer that the outsource provider make no substantive decisions about what licenses are acceptable, and leave those decisions to the customer. But assuming that some customers will require assistance making that decision—or at least appreciate the outsourcer identifying issues early rather than late—outsource providers also need to understand the policies their customers are likely to have.

III. PERMISSIVE VS. "FREE" SOFTWARE

The first thing any lawyer learns about open source is that there are two kinds: permissive or "open" software, and "free software." Terminology in the open source world is inconsistent and fraught with political peril, but the two categories are essential to understanding open source compliance. Outsource providers need to understand this because their clients will react to the use of these two types of open source software very distinctly.

A. Permissive Licenses

This type of license includes Apache, BSD, and MIT licenses.[4] These licenses have few requirements—essentially limited to notice requirements—and code received under these licenses can be included in proprietary products.

B. Hereditary Licenses

This type of license includes GPL, LGPL, Mozilla (MPL), CDDL, CPL, and Eclipse.[5] These are sometimes called "free software," "reciprocal," or "copyleft" licenses—or more pejoratively, "viral" licenses—but members of the open source community will argue about which epithets apply to which licenses.[6] The essential

4. The text of these licenses, in their standard forms, is available at http://www.opensource.org/licenses.

5. *Id.*

6. I use the term "hereditary" because it is the most neutral, and, I feel, the more accurately descriptive. But this term is not in common use. The Free Software Foundation calls this category "free software," but that description does not always include so-called "weak copyleft" licenses like Mozilla Public License and Eclipse. *See* The Free Software Foundation, Various Licenses and Comments About Them, http://www.gnu.org/philosophy/license-list.html (last visited Apr. 11, 2008). The former category is variously called permissive, BSD-type, or simply open source, and its terminology tends to be a less sensitive subject because the precept behind such licenses is the exercise of minimal downstream control. *See, e.g.*, The Apache

feature of these licenses is that if you distribute the code provided under these licenses, or modifications of the code, you must distribute it in source code form under the terms of that license.[7] The scope of modifications covered, alternatives for licensing of binary code, and other terms vary by license. However, most clients of outsource development providers will have more restrictive policies for use of code under hereditary licenses than for permissive ones.

IV. TYPES OF CUSTOMERS AND PRODUCTS

The outsource provider's customers will be more or less sensitive to the use of open source in its development depending on the type of product being developed and the nature of the customer's business. This is because some code is more likely to be distributed than other code. As a general rule, the copyleft obligations of hereditary licenses like GPL are not invoked until distribution of a product occurs.[8] What constitutes distribution can be a thorny issue,[9] but generally an on-line service that provides users with access to the functionality of the software, but not a copy of the software itself, is not distribution.[10] Therefore, development of software to operate online services, or software to be marketed under a SAAS model, will be less sensitive to copyleft concerns. However, lack of distribution is not a panacea for these issues. Online services and SAAS products can easily morph into distributed software solutions. Also, the gray areas around what constitutes distribution can mean that providing copies of code to affiliates can trigger copyleft obligations.

Software Foundation, How the ASF Works, http://www.apache.org/foundation/how-it-works.html (last visited Apr. 11, 2008).

7. *See, e.g.*, The Free Software Foundation, GNU General Public License, Version 2, § 2(b), http://www.gnu.org/licenses/gpl-2.0.html (last visited Apr. 11, 2008).

8. The exception that proves the rule is the Affero GPL, recently released in a new version stewarded by the Free Software Foundation. *See* Fabrizio Capobianco, AGPL is OSI approved. Sweet Victory (Mar. 13, 2008), http://www.funambol.com/blog/capo/2008/03/agpl-is-osi-approved-sweet-victory.html (discussing why the distribution threshold for triggering copyleft is important).

9. See the discussion below of whether the use of outsource providers constitutes distribution.

10. This conclusion may be more reliable under U.S. law than other law. Some open source licenses refer to the copyright term "distribution," but others, most notably GPL version 3, refer to analogous terms under local copyright law. Thus, the activities that trigger copyleft obligations may be broader, particularly under German law. This topic is beyond the scope of this article.

V. WHITE LISTS, BLACK LISTS, AND GRAY LISTS

Customer policies often come in the form of lists of licenses that are prohibited, allowed, and allowed on approval—otherwise known as White Lists, Black Lists, and Gray Lists. The more sophisticated customers will have developed written policies that identify these categories, and describe the business processes for submitting Gray List items for review. Although policies vary by industry, product type, anticipated use, and individual preference, a typical policy might contain the information in the following table.

Note that this kind of a table implies that the customer has made many legal conclusions about the meaning of these licenses, and reasonable people—even reasonable lawyers—can differ on these conclusions. The conclusions reflected below are just examples. Some companies, in particular, may not place GPL/LGPL v.3 on their Black Lists, and some may place all versions of GPL on their Black Lists. In the table below, "proprietary" refers to code that is licensed under a conventional, binary or object code only (not an open source) license.

License	Compliance Rules	Comments
GPL version 2	• No proprietary code may be linked (dynamically or statically) to GPL code. • Proprietary applications may run in user space on top of Linux, even though Linux is covered by GPL. • GPL code may interface with proprietary code through pipes, sockets, data files, or fork/exec calls.	• This generally follows the Free Software Foundation's interpretations and guidance, though there are gray areas around the margins
GPL version 3	• Do not use	
LGPL version 3	• Do not use	
LGPL version 2.1	• LGPL code may not be used by proprietary code except via dynamically linked libraries. • LGPL code may not contain macros or in-line functions of over 10 lines.	• Legal must confirm that license for proprietary code does not prevent modification or reverse engineering to the extent

License	Compliance Rules	Comments
		necessary to allow exercise of LGPL rights for the library code, including modification of library and re-build of product with modified library. • Legal must confirm no patent issues
GPL v.2 + Special Exception	• GPL + Exception code may be used if it is an unmodified library	• GPL + Exception allows proprietary code to be linked, dynamically or statically. Therefore unmodified code is unlikely to violate compliance rules. • Legal must confirm no patent issues
Mozilla 1.1	• Proprietary code may be combined with MPL code if it is in separate source files • MPL code may be used if it is an unmodified library	• Legal must confirm no patent issues • Note that some companies will place Mozilla 1.1 on the Black List due to patent termination provisions; this may not apply to Mozilla 1.0
CDDL	• Proprietary code may be combined with CDDL code if it is in separate source files • CDDL code may be used if it is an unmodified library	• Legal must confirm no patent issues • Note that some companies will place CDDL on the Black List due to

License	Compliance Rules	Comments
		patent termination provisions
Eclipse	• Eclipse code may be used if it is an unmodified library	• Legal must confirm no patent issues
CPL	• CPL code may be used if it is an unmodified library	• Legal must confirm no patent issues
Apache 2.0	• OK	• Legal must confirm no patent issues
BSD	• OK	
MIT	• OK	
Apache 1.1	• OK	
Apache 1.0	• Do not use	• Note that some companies will place Apache 1.0 on the Black List due to its advertising clause, since deprecated in favor of Apache 1.1

VI. DEVELOPMENT TOOLS

The table above represents an illustration of a policy for use of code to be integrated into a product. But the rules for development tools are quite different. Unfortunately, there is some confusion over what is referred to as a development tool.

When people refer to a development tool they usually mean a compiler, development environment, SDK, or text editor. There are some gray areas, though. For instance, PERL may refer to the PERL interpreter (which is dual licensed under GPL and the Artistic License). However, PERL is also a language, and so there are PERL libraries and development environments available under various terms. The fact that the PERL engine is licensed under GPL will not affect the intellectual property rights in PERL scripts that are processed using that engine. However, using PERL scripts covered by GPL may affect the licensing of accompanying or modified scripts.

SDKs and compilers may also raise compliance issues. These utilities may incorporate code snippets or run-time libraries into a program. If so, it is the licensing of the code or libraries that matters, not the licensing of the SDK or compiler. Thus, GCC (the GNU C Compiler) is licensed under GPL, but the standard libraries are licensed under GPL + exception, allowing them to be incorporated into proprietary programs.[11]

One should take particular care with the term "API," which is used inconsistently as a programming interface description (i.e. documentation) and a set of code libraries for implementing a particular device, platform, or other technology. In the latter case, the API includes software code. In the former, it probably does not. Things can get particularly confusing when non-software is licensed under a software license like GPL. This generally reflects confusion on the part of the licensor, and is not common among the bigger, better-managed projects—which usually use the GNU Free Documentation License or a Creative Commons license for documentation.

The bottom line is that a development tool licensed under GPL that does not inject any code into the program being developed does not cause compliance issues for the product being developed, because the GPL code itself is never part of the product. Examples would include debuggers, text editors, bug trackers and concurrent versioning systems.

A language interpreter licensed under GPL does not generally cause GPL compliance issues[12]—even though it is possible that standard system libraries may be covered by GPL and may need to link to code being interpreted in order to run. PERL, Python, and Java are examples of language interpreters.

---

11. For the Free Software Foundation's explanation, see The Free Software Foundation, Frequently Asked Questions about the GNU Licenses, http://www.gnu.org/licenses/gpl-faq.html#CanIUseGPLToolsForNF (last visited Apr. 11, 2008) (answering the question "can I use GPL-covered editors such as GNU Emacs to develop non-free programs. Can I use GPL-covered tools such as GCC to compile them?").

12. For the Free Software Foundation's view, see The Free Software Foundation, Frequently Asked Questions about the GNU Licenses, http://www.gnu.org/licenses/gpl-faq.html#IfInterpreterIsGPL (last visited Apr. 11, 2008) (answering the question "if a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses?").

VII. THE "DEVELOPER PROBLEM"

The most complex open source question for outsource developers is whether use of an outside developer constitutes a copyleft-triggering distribution. The answer to this question is fact-specific and can change based on facts that may not seem to change the business case much. The conceptual difficulty with this question is that, if you are an outsource developer, your customers will likely perceive you as equivalent to an in-house developer, so the idea that engaging you to perform development may change the customer's outbound licensing posture will seem counterintuitive to them.

The simplest case is when your customer engages you to develop new programs that the customer intends to release under an open source license. In this case, there are no restrictions of third party inbound licenses to consider.[13] Presumably, your customer will own the code via an assignment of rights. Thus, your customer has complete discretion to release that code via an open source license or not. Whether you distribute the code to your customer when you deliver it is irrelevant.

A harder case occurs when your customer engages you to perform development of modifications to code covered by a license like GPL. If your customer owns the rights to the code, and merely happens to have already released it under GPL, this case is the same as the one described above. By definition, the owner of the code has no restrictions of third party inbound licenses to consider. However, if your customer owns the rights to only some of the code (if, for instance, it has used community developments licensed under GPL) or has modified third party GPL code before giving it to you for further development, the question arises as to whether the customer's distribution to you invokes the copyleft requirements of GPL, or your delivery of the finished product to the customer invokes these obligations.

The Free Software Foundation's FAQ on the GPL says:

> *Does the GPL allow me to develop a modified version under a nondisclosure agreement?*
>
> Yes. For instance, you can accept a contract to develop changes and agree not to release your changes until the client says ok. This is permitted because in this case no GPL-covered code is being distributed under an NDA.

13. This is not quite true if you are developing plug-ins to GPL code, which is discussed below.

> You can also release your changes to the client under the GPL, but agree not to release them to anyone else unless the client says ok. In this case, too, no GPL-covered code is being distributed under an NDA, or under any additional restrictions.
>
> The GPL would give the client the right to redistribute your version. In this scenario, the client will probably choose not to exercise that right, but does have the right.[14]

But note that this question appears to apply to modifications of GPL code that was originally developed by the same developer. The FAQ also says:

> *Is making and using multiple copies within one organization or company "distribution"?*
>
> No, in that case the organization is just making the copies for itself. As a consequence, a company or other organization can develop a modified version and install that version through its own facilities, without giving the staff permission to release that modified version to outsiders.
>
> However, when the organization transfers copies to other organizations or individuals, that is distribution. In particular, providing copies to contractors for use off-site is distribution.[15]

The FAQ does not mention whether "contractors" in this case includes developers or other kinds of contractors, but it does suggest that delivery of GPL code to a contractor for further development triggers the copyleft obligations of GPL.

There are legitimate questions, however, as to whether the Free Software Foundation's interpretation of the GPL is legally binding.[16] And there are arguments that providing code to a contractor acting on one's behalf is not distribution. As a general matter of agency law, if the contractor is acting on behalf of the company, then the two would be considered separate entities between which a transfer could take place. Also, to the extent the U.S. copyright law defines distribution,

---

14. The Free Software Foundation, Frequently Asked Questions about the GNU Licenses http://www.gnu.org/licenses/gpl-faq.html#DevelopChangesUnderNDA (last visited Apr. 11, 2008). This FAQ now presumably is addressed toward GPL version 3, but the answer is not written as specific to version 3.

15. The Free Software Foundation, Frequently Asked Questions about the GNU Licenses, http://www.gnu.org/licenses/gpl-faq.html#InternalDistribution (last visited Apr. 11, 2008).

16. This is a complex topic and beyond the scope of this article. For further discussion, *see* HEATHER J. MEEKER, THE OPEN SOURCE ALTERNATIVE: UNDERSTANDING RISKS AND LEVERAGING OPPORTUNITIES 223-32 (2008).

it requires making a copy publicly available, and so delivery to a contractor may not qualify.

The bottom line on this issue is that it is unsettled. So, before delivery by a customer to an outsource provider of third party GPL code to be modified, or before delivery by an outsource provider to a customer of third party GPL code that has been modified, each should stop to consider whether the delivery is possible outside the scope of GPL, and within the confines of the outsource development agreement.

VIII. NOTICES

All open source licenses have copyright notice or attribution requirements. If a product contains many open source modules, it may be quite difficult to comply with all of them simultaneously. For instance, some may require notices in user documentation and your customer may not provide user documentation. But at a minimum, if your customer plans on distributing copies of the code you have developed, your customer will need to gather all the notices for all open source software used in the product. Because of the operational difficulty in adhering to different notice provisions, many companies develop a general policy for complying with notice issues—such as posting notices on a Web site accessible by customers or delivering notices via technical bulletins. Although these policies often are not technically compliant with each open source license, they are intended to address the spirit of the notice and attribution requirements. Gathering the notices to be included for open source can be time consuming, so it is best to gather them in a central source as you perform development. Doing so will save your customer the time and effort of backtracking.

IX. HELPING YOUR CUSTOMERS HELP THEMSELVES

The brief summary in this article of issues for outsource development services providers is, in a way, a crash course in open source compliance for any company doing development. Of course, if your are lucky enough that your customers tell you exactly how they want you to handle open source in your work for them, you can simply follow their direction. But for customers that are still sorting out their policies on open source, outsource providers can add value by assisting and guiding their customers. By doing this, outsource providers can demonstrate a level of sophistication that will give them

a competitive edge, grow with their customers, and together, come of age in the open source world.

* * *